AMENDMENTS TO THE CLAIMS

Kindly replace the claims as follows.

1. (original) A method, comprising the steps of:

during execution of an instruction on a computer, in response to an operation of the instruction calling for an architecturally-visible side-effect in an architecturally-visible storage location, storing a value representative of an architecturally-visible representation of the side-effect, a format of the representative value being different than an architecturally-visible representation of the side-effect, and resuming the execution without generating the architecturally-visible side-effect;

later writing the architecturally-visible representation corresponding to the representative value into the architecturally-visible storage location.

2. (original) The method of claim 1, wherein the later writing is triggered by a read of the architecturally-visible storage location.

3. (original) The method of claim 1, wherein the later writing is triggered by the completion of the execution of the instruction.

4. (original) The method of claim 1, wherein the architecturally-visible storage location includes at least two of a floating-point instruction pointer, a floating-point data pointer, and a floating-point opcode.

5. (original) The method of claim 1, wherein the representative value is held in a temporary register until several other side-effects are also ready to be committed to the architecturally-visible storage location simultaneously.

6. (original) The method of claim 1, wherein the representative value is held in a non-addressable storage register, and a process of the instruction only cedes control on an instruction boundary, so that the non-addressable information is not lost.

7. (original) The method of claim 1,

context switch software programmed to store a context of a first process and to load a context of a second process to place the second process into execution, each context comprising a set of resources to be reloaded whenever a process associated with the context is reloaded for execution;

a multi-stage execution pipeline of a computer, at least some instructions executed in the pipeline maintaining results in storage resources outside the context resource set, instructions for execution by the pipeline being marked to indicate whether or not a context switch may be performed at a boundary of the marked instruction.

8. (original) The method of claim 1,

during hardware execution of an instruction stream including the instruction, recognizing a condition that is a superset of a condition being monitored for occurrence, and raising a first exception as a result of recognizing the superset condition;

in software, filtering the superset condition to determine whether the monitored condition has occurred;

if the monitored condition has occurred, establishing a second exception to be raised after execution of further instructions of the instruction stream.

9. (original) A computer, comprising:

circuitry designed to recognize an operation of an instruction calling for an architecturally-visible side-effect in an architecturally-visible storage location, and in response, to store a value representative of an architecturally-visible representation of the side-effect, a format of the representative value being different than an architecturally-visible

representation of the side-effect, and to resume the execution without generating the architecturally-visible side-effect;

circuitry and/or software designed to later write the architecturally-visible representation corresponding to the representative value into the architecturally-visible storage location.

10. (original) The computer of claim 9, further comprising:

circuitry designed to recognize a condition in which a second instruction is to affect the execution of a third instruction, and in response, to set the processor into single-step mode;

circuitry designed to raise a single-step exception after executing the second instruction, and to set the processor out of single-step mode.

11. (original) The computer of claim 9, wherein the storage location is a location in main memory or a cache memory of the computer.

12. (original) The computer of claim 9, wherein the storage location is a general purpose register of the computer having no address in an address space of the computer.

13. (original) A method, comprising the steps of:

storing a context of a first process and loading a context of a second process to place the second process into execution, each context comprising a set of resources to be reloaded whenever a process associated with the context is reloaded for execution;

at least some instructions executed in a multi-stage execution pipeline of the computer maintaining results in storage resources outside the context resource set, instructions for execution by the pipeline being marked to indicate whether or not a context switch may be performed at a boundary of the marked instruction.

14. (original) The computer of claim 13, comprising:

recognizing a condition that is a superset of a condition being monitored for occurrence, and raising a first exception as a result of recognizing the superset condition;

in software, filtering the superset condition to determine whether the monitored condition has occurred;

if the monitored condition has occurred, establishing a second exception to be raised after execution of further instructions of the instruction stream.

15. (original) The computer of claim 13, comprising:

recognizing a condition in which an instruction is to affect the execution of a second instruction, and in response, setting the processor into single-step mode;

taking a single-step exception after executing the second instruction, and setting the processor out of single-step mode.

16. (original) A computer, comprising:

context switch software programmed to store a context of a first process and to load a context of a second process to place the second process into execution, each context comprising a set of resources to be reloaded whenever a process associated with the context is reloaded for execution;

a multi-stage execution pipeline of a computer, at least some instructions executed in the pipeline maintaining results in storage resources outside the context resource set, instructions for execution by the pipeline being marked to indicate whether or not a context switch may be performed at a boundary of the marked instruction.

17. (original) The computer of claim 16, comprising:

circuitry designed to recognize an operation of an instruction calling for an architecturally-visible side-effect in an architecturally-visible storage location, and in response, to store a value representative of an architecturally-visible representation of the side-effect, a format of the representative value being different than an architecturally-visible

representation of the side-effect, and to resume the execution without generating the architecturally-visible side-effect;

circuitry and/or software designed to later write the architecturally-visible representation corresponding to the representative value into the architecturally-visible storage location.

18. (original) The computer of claim 16, wherein the instructions are marked by a marker on a last of the instructions generated by decoding an external-form instruction fetched from memory.

19. (original) The computer of claim 18, wherein the context switch is triggered in response to an action of a non-final one of the instructions generated by decoding.

20. (original) The computer of claim 16, wherein the instructions are marked by a marker on an intermediate one of the instructions generated by decoding an external-form instruction fetched from memory, the marker indicating an iteration boundary of an external-form instruction specifying repeated execution of an operation.

21. (currently amended) The computer of claim 16, further comprising the step of:

based on the memory protection check of a ~~the~~ younger of the instructions executing in the pipeline, canceling the effect of an ~~the~~ older of the ~~two~~ instructions.

22. (currently amended) A method, comprising the steps of:

during hardware execution of an instruction stream, recognizing a condition that is a superset of a condition whose occurrence is desired to be detected, and raising a first exception as a result of recognizing the superset condition;

in software, filtering the superset condition to determine whether the desired condition has occurred, and if so, gathering further information about the condition;

if the desired condition is determined to have occurred, establishing a second exception to be raised after execution of further instructions of the instruction stream, and making the further information available for a handler of the second exception.

23. (original) The method of claim 22, comprising:

recognizing a condition in which an instruction is to affect the execution of a second instruction, and in response, setting the processor into single-step mode;

taking a single-step exception after executing the second instruction, and setting the processor out of single-step mode.

24. (currently amended) A computer, comprising:

an instruction execution pipeline designed to execute instructions, and to monitor the executing instructions for a condition arising during execution that is a superset of a condition whose occurrence is desired to be detected, and to raise a first exception as a result of recognizing the superset condition;

software designed to filter the superset condition to determine whether the monitored condition has occurred, and if the monitored condition is determined to have occurred, to gather further information about the condition, and to establish a second exception to be raised after execution of further instructions of the instruction stream, and to make the further information available for a handler of the second exception.

25. (original) The computer of claim 24, further comprising:

circuitry designed to recognize an operation of an instruction calling for an architecturally-visible side-effect in an architecturally-visible storage location, and in response, to store a value representative of an architecturally-visible representation of the side-effect, a format of the representative value being different than an architecturally-visible representation of the side-effect, and to resume the execution without generating the architecturally-visible side-effect; and

circuitry and/or software designed to later write the architecturally-visible representation corresponding to the representative value into the architecturally-visible storage location.

26. (original) The computer of claim 24:

further comprising context switch software programmed to store a context of a first process and to load a context of a second process to place the second process into execution, each context comprising a set of resources to be reloaded whenever a process associated with the context is reloaded for execution;

wherein at least some instructions executed in the pipeline maintaining results in storage resources outside the context resource set, instructions for execution by the pipeline being marked to indicate whether or not a context switch may be performed at a boundary of the marked instruction.

27. (original) The computer of claim 24, wherein the desired condition is a memory reference to a narrow range of addresses, and the superset condition is a memory reference to a broader range of addresses.

28. (original) The computer of claim 27, wherein the broader range of addresses is a cache line.

29. (original) The computer of claim 24, wherein the monitored condition is a memory reference to an address of a reference class, and the superset condition is a memory reference to the address, without respect to reference class.

30. (original) The computer of claim 24, wherein the filtering software records the nature of the monitored condition that has occurred.

31. (original) The computer of claim 24, wherein the filtering software records multiple occurrences of desired conditions before the second exception is raised.

32. (original) The computer of claim 24, wherein the second exception vectors to a debug entry point of an operating system.

33. (original) The computer of claim 24, wherein the condition is an exception recognized on one of a plurality of instructions generated by a single instruction fetched from a memory, and the second exception is deferred until an instruction boundary of the instruction fetched from memory.

34. (original) A method, comprising the steps of:

during execution of a program on a computer, recognizing a condition in which an instruction is to affect the execution of a second instruction, and in response, setting the processor into single-step mode;

taking a single-step exception after executing the second instruction, and setting the processor out of single-step mode.

35. (original) The method of claim 34, further comprising the steps of:

in response to an operation of the instruction calling for an architecturally-visible side-effect in an architecturally-visible storage location, storing a value representative of an architecturally-visible representation of the side-effect, a format of the representative value being different than an architecturally-visible representation of the side-effect, and resuming the execution without generating the architecturally-visible side-effect;

later writing the architecturally-visible representation corresponding to the representative value into the architecturally-visible storage location.

36. (original) A computer, comprising:

hardware designed to recognize a condition rising during execution of an instruction on a computer, in which the instruction is to affect the execution of a second instruction;

hardware and/or software designed to respond to the recognizing by setting a processor of the computer into single-step mode; and

hardware and software designed to respond to execution of the second instruction by setting the computer out of single-step mode.

37. (original) The computer of claim 36, further comprising:

context switch software programmed to store a context of a first process and to load a context of a second process to place the second process into execution, each context comprising a set of resources to be reloaded whenever a process associated with the context is reloaded for execution;

a multi-stage execution pipeline of a computer, at least some instructions executed in the pipeline maintaining results in storage resources outside the context resource set, instructions for execution by the pipeline being marked to indicate whether or not a context switch may be performed at a boundary of the marked instruction.

38. (original) The computer of claim 36, further comprising:

an instruction execution pipeline designed to execute instructions, and to monitor the executing instructions for a condition arising during execution that is a superset of a condition whose occurrence is desired to be detected, and to raise a first exception as a result of recognizing the superset condition;

software designed to filter the superset condition to determine whether the monitored condition has occurred, and if the monitored condition is determined to have occurred, to establish a second exception to be raised after execution of further instructions of the instruction stream.

39. (original) The computer of claim 36, wherein the first instruction writes a stack segment register.

40. (original) The computer of claim 36, wherein the first instruction and second instructions are generated by an instruction decoder in response to a single instruction fetched from a memory.

41. (original) The computer of claim 40, wherein the first instruction writes to a group of interrupt flags.

42. (original) The computer of claim 36, wherein the first and second instructions are generated by a decoder for a complex instruction set in response to decoding a single instruction of the complex instruction set.

43. (original) The computer of claim 42, wherein the second instruction lies on a boundary between iterations of a loop executed within the single complex instruction.

44. (original) The computer of claim 36, wherein servicing a single-step exception includes querying a debug touch record.

45. The computer of claim 36, wherein the first instruction writes an interrupt enable flag of the computer.